Michele Marchese Andreu
C21741655

Game Engines project documentation

Michele Marchese Andreu

C21741655

YouTube video:

Link to the video: https://youtu.be/Es6QX_h6WTI

Research:

The research on the ducks is the only one that I made in person due to the limitations I had. It was impossible for me to do research on whales, clownfish, or anglerfish in person.

Ducks:

To study the ducks, I went to phoenix park and I took a couple of pictures and some videos.

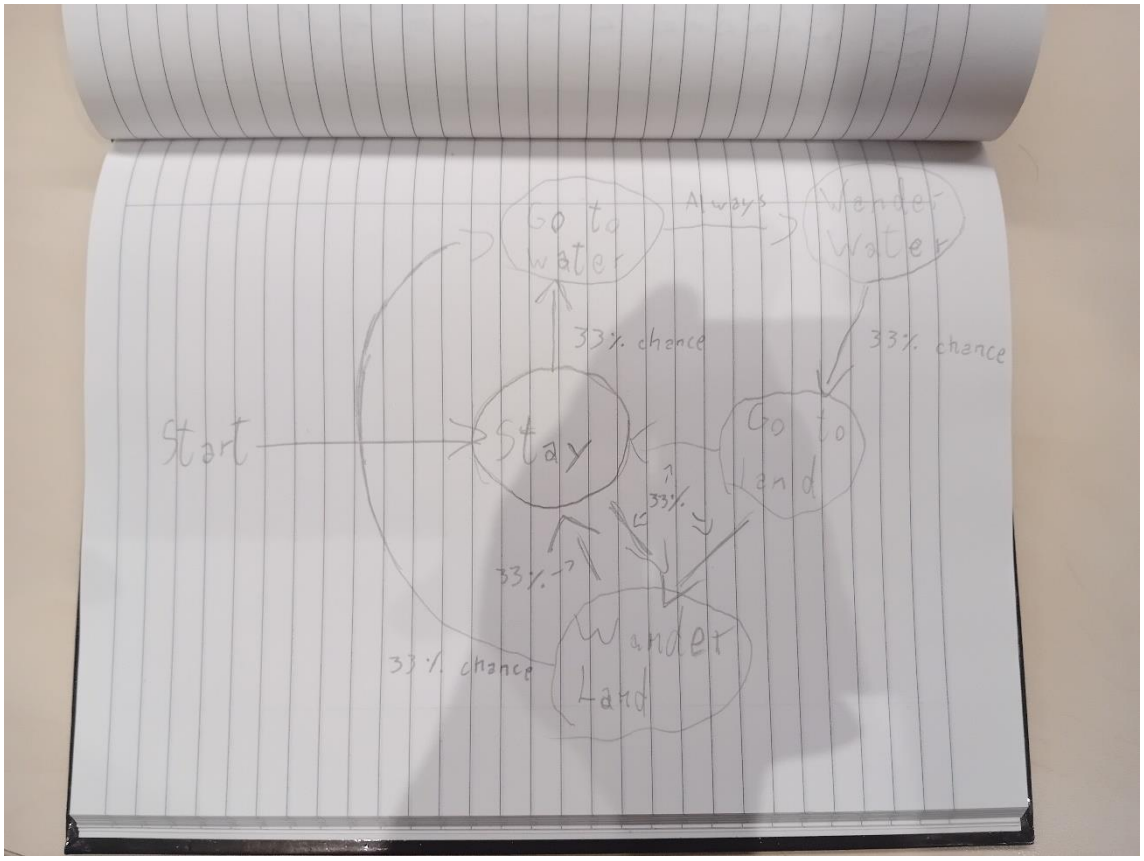
(the videos can be found in the folder called Duck Videos)

After studying them for a while I found that they had 5 different behaviours:

- Stay: This state is when the ducks are on land and they sit and stay still. This behaviour is exclusive to when they are on land because they would not stay still while on water.
- Go to water: This behaviour is for when they are walking on land in the direction of water with the intention of going into it.
- Wander water: This state is for when the duck is already on water and it starts to wander.

- Go to land: This behaviour is for when the duck is on water and goes in the direction of land with the intention of getting out of the water.
- Wander Land: This state is for when the duck is on land and t walks in a random direction.

They seemed to switch between these states at random while they were either on land or water, So I ended up making this flow-chart:



To simulate the duck in unity I used 2 behaviours: Seek and Wander Noise. They were implemented in the following ways:

- Stay: The seek behaviour and noise wander behaviours weight is 0 and the target object for seek is set to null
- Go to water: The wander noise has a weight of 0 and seek a weight of 1. The target object would be 1 of 4 objects placed on the water and it is set to be the nearest one to the duck at the moment of the switch.
- Wander water: Wander noise weight is set to 7 and seek weight is set to 1. The target object is set to whatever the previous target object was in the “Go to water” state.
- Go to land: The wander noise has a weight of 0 and seek a weight of 1. The target object is the centre of the island.
- Wander land: Wander noise weight is set to 7 and seek weight is set to 1. The target object is the centre of the island.

With these behaviours and the flowchart I was able to implement the behaviour of ducks very similarly to real life. The main differences are that they don't switch behaviours as frequently (which I kept for the sake of having a better viewing experience), that they always go to a specific point in the water instead of a random spot and finally that they don't go in pairs. While studying they would very rarely move in pairs while on the water, but it was so rare that I decided to not implement it (although it sometimes looks like it is implemented).

The duck ended up being the most complex boid in the simulation.

Whale:

The whale was the second boid I implemented.

The first thing I did for the whale was code the tail and fin movement. The whale, unlike other aquatic animals, moves its tail up and down because it is a mammal and their bones naturally blend up and down. Information taken from the whale and dolphin conservation centre:

(https://uk.whales.org/whales-dolphins/why-do-whale-and-dolphin-tails-go-up-and-down/#:~:text=Unlike%20fish%2C%20whales%20and%20dolphins,for%20propulsion%20through%20the%20water)

To replicate this, I created a sin wave movement code that would take an array of transforms and would move them up and down depending on the time, an amplitude, an angle, a frequency ad their position on the array. This recreated the movement perfectly. As for the fins the rotate on all of their axis depending on a sin wave as well.

The majority of recorded movement of whales is recorded on a large scale, partially due to their size, it has been studied that whales will travel to warmer waters for skin maintenance.

(see this link: https://www.fisheries.noaa.gov/feature-story/why-do-whales-migrate-they-return-tropics-shed-their-skin-scientists-say#:~:text=%E2%80%9CInstead%20of%20whales%20migrating%20to,there%2C%E2%80%9D%20the%20scientists%20wrote)

After studying various videos, I noticed that they tend to wander in short distance travel.

(see this video: https://www.youtube.com/watch?v=Uvd2UYS8wVs)

Just making the whale wander with noise or jitter wander would be a problem due to its size, so instead I made it follow a path that circles around the island. But only follow a path would not feel natural so I created 2 states:

- Follow path: In this state the whale will follow the path continuously without being interrupted. The follow path behaviour has a weight of 1 and the noise wander has a weight of 0.

- Wander: In this state the whale will use a noise wander movement while also following the path. The follow path behaviour has a weight of 1 and the noise wander has a weight of 2.

The follow path has a 40% chance to transition to the wander behaviour every 3 seconds. Meanwhile, the wander has a 60% to transitions to follow path every 3 seconds.

Comparing it to the real movement it is not as random because it follows a path, but it was necessary to make the simulation more concentrated. It was also impossible to replicate the long distance travel of whales for the same reason.

Baby Whale:

The baby whale, as its name implies, is a boid that represents the offspring of the bigger whale. While they may share the same model with the only difference being its scale, their behaviours are very different.

Baby whales (as seen in this video: https://www.youtube.com/watch?v=pJqlkuVKO2k) stay near their mother at all times and tend to wander a bit but not too far away from their mother.

To replicate this, I used 2 different behaviours: Pursue and Obstacle avoidance. For the pursue script the target is always set as the big whale and obstacle avoidance's layers to avoid are set as the layer for the whale and the island.

It has 2 different states:

- Purse: In this state it pursues the big whale constantly. The pursue weight is set to 4 and the avoidance weight is set to 0.
- Avoidance: In this state it will avoid both the whale and the island constantly. The pursue weight is set to 0 and the avoidance weight is set to 17.

The baby whale will switch from the pursue state to the avoidance when either the whale or the island enter a sphere trigger that is around the game object. It will switch from the avoidance state to pursue once either the whale or the island exit the trigger.

This state machine was created for 2 reasons. The first one is to better control how close the baby whale can get to the big whale and the island and the second one is to prevent the baby whale from clipping into the model of the big whale, which was a bug that happened before implementing the state machine.

Comparing it to real life it is for the most part the same, the only thing I wasn't able to replicate is the baby whale touching the big whale due to it clipping into the model and preferring bug prevention over behaviour fidelity.

Fish:

The fish boid is primarily based on the clownfish due to their connection to anemone. In this project anemone are look more like algae to improve the performance of the game.

They are related due to their symbiotic relationship that helps both of them by the anemone providing shelter to the fish and the fish cleaning the anemone and scaring off their predators. This relationship is famously portrayed in the movie nemo.

(see more here: https://asknature.org/strategy/intricate-relationship-allows-the-other-to-flourish/)

The clownfish tends to stay either near the anemone or inside of it.

(see more: https://www.youtube.com/watch?v=uCLW06xQTIs)

To represent this, I created 2 behaviours with seek and 2 different harmonic movements, a vertical one and a horizontal one:

- Seek: This state has all of the behaviours active and the seek target object is set to the centre of the anemone.
- Not seek: In this state both harmonic moments are active, but the seek behaviour weight is set to 0.

The way they switch between each other is very simple, once the fish gets to the centre of the anemone it will switch to not seek to move in a straight line and then, after waiting for 2 seconds it will automatically switch back to the seek state.

The fish has a 3rd state that overrides both the other states that is the Angler state:

- Angler: In this state the fish will go into the mouth of the anglerfish and once it gets to it will delete itself.

The way it transitions to this state is by entering a trigger that the anglerfish has.

Comparing it to real life it has 2 major differences, the first one is that it cannot stay still inside the anemone nor clean it, which is what clownfish does most of the time, and the second one is that it will never interact with anglerfish in real life so we don't know if it would go to the light produced by it.

Anglerfish:

There is very little documented evidence about anglerfish, but it is known that they don't swim very fast and don't actively hunt, they just wait until a prey comes nearby.

(see more here: https://www.youtube.com/watch?v=VqPMP9X-89o)

To represent this the it has 1 state that uses the seek behaviour and the noise wander behaviour. The seek target object is set to the nearest anemone at the beginning and then every 20 seconds it has a random chance to switch it to the nearest anemone.

Comparing it to real life we don't really know if it would behave very differently due to the lack of research, but we know for sure that it would never interact with clownfish specifically.

Camera:

Michele Marchese Andreu
C21741655

The camera follows one of 5 paths depending on which animal you select in the UI. If you select either the whale or one of the anglerfish, it will get attached to game object that is part of the boid itself, not use a path follow. Additionally it will change the colour of the screen depending on if it is underwater or not.

What did I learn?

The main thing I learnt is to code state machines. On previous projects when I had to code state machines for AI I would use only 1 script that would contain all of the states and would switch between them exclusively using if and else if statements. Now I learnt to use abstract classes to create various scripts that derive from that class and then use a separate MonoBehaviour to switch call the other scripts and run one at a time. This helped with how clean the code was and with the efficiency of the code because it only had to check 1 or 2 if statements at the same time instead of 7 or 8.

I also learnt to better study the behaviours of real animals to recreate them using artificial intelligence. In my research I found that most of them tend to wander in a specific zone, to recreate this I had to use the seek behaviour and the wander noise with seek having a higher weight than wander noise. Then for specific attributes I used different behaviours suck as pursue, harmonic movement or path follow. With all of this I was able to almost perfectly recreate most animals' real life behaviour.

Assets I did not create:

Music: it was created by **JuliusH** on Pixaby with the title **Deep Water Adventure - Ambient Underwater Music** (link: https://pixabay.com/music/ambient-deep-water-adventure-ambient-underwater-music-167787/).

Skybox: taken from the unity package "Fantasy skybox FREE".

Mountain: taken from the unity asset package "Polygon Farm".

Rainbow gradient image: taken from Wikimedia (link: https://commons.wikimedia.org/wiki/File:Rainbow-gradient-fully-saturated.svg) but I rotated it 90 degrees.